

nag_real_apply_q (f01qdc)

1. Purpose

nag_real_apply_q (f01qdc) performs one of the transformations

$$B := QB \quad \text{or} \quad B := Q^T B,$$

where B is an m by $ncolb$ real matrix and Q is an m by m orthogonal matrix, given as the product of Householder transformation matrices.

This function is intended for use following **nag_real_qr (f01qcc)**.

2. Specification

```
#include <nag.h>
#include <nagf01.h>
```

```
void nag_real_apply_q(MatrixTranspose trans, Nag_WhereElements wheret,
    Integer m, Integer n, double a[], Integer tda, double zeta[],
    Integer ncolb, double b[], Integer tdb, NagError *fail)
```

3. Description

Q is assumed to be given by

$$Q = (Q_n Q_{n-1} \dots Q_1)^T,$$

Q_k being given in the form

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix},$$

where

$$T_k = I - u_k u_k^T$$

$$u_k = \begin{pmatrix} \zeta_k \\ z_k \end{pmatrix},$$

ζ_k is a scalar and z_k is an $(m - k)$ element vector. z_k must be supplied in the $(k - 1)$ th column of **a** in elements **a**[k][$k - 1$], ..., **a**[$m - 1$][$k - 1$] and ζ_k must be supplied either in **a**[$k - 1$][$k - 1$] or in **zeta**[$k - 1$], depending upon the parameter **wheret**.

To obtain Q explicitly B may be set to I and premultiplied by Q . This is more efficient than obtaining Q^T .

4. Parameters

trans

Input: the operation to be performed as follows:

trans = **NoTranspose**, perform the operation $B := QB$.

trans = **Transpose** or **ConjugateTranspose**, perform the operation $B := Q^T B$.

Constraint: **trans** must be one of **NoTranspose**, **Transpose** or **ConjugateTranspose**.

wheret

Input: indicates where the elements of ζ are to be found as follows:

wheret = **Nag_ElementsIn**, the elements of ζ are in **a**.

wheret = **Nag_ElementsSeparate**, the elements of ζ are separate from **a**, in **zeta**.

Constraint: **wheret** must be **Nag_ElementsIn** or **Nag_ElementsSeparate**.

m

Input: m , the number of rows of A .

Constraint: **m** $\geq$ **n**.

n

Input: n , the number of columns of A .
 When **n** = 0 then an immediate return is effected.
 Constraint: **n** $\geq$ 0.

a[m][tda]

Input: the leading m by n strictly lower triangular part of the array **a** must contain details of the matrix Q . In addition, when **wheret** = **Nag_ElementsIn**, then the diagonal elements of **a** must contain the elements of ζ as described under the parameter **zeta** below.
 When **wheret** = **Nag_ElementsSeparate**, the diagonal elements of the array **a** are referenced, since they are used temporarily to store the ζ_k , but they contain their original values on return.

tda

Input: the second dimension of the array **a** as declared in the function from which nag_real_apply_q is called.
 Constraint: **tda** $\geq$ **n**.

zeta[n]

Input: if **wheret** = **Nag_ElementsSeparate**, the array **zeta** must contain the elements of ζ . If **zeta**[$k - 1$] = 0.0 then T_k is assumed to be I otherwise **zeta**[$k - 1$] is assumed to contain ζ_k .
 When **wheret** = **Nag_ElementsIn**, **zeta** is not referenced and may be set to the null pointer, i.e., (double *)0.

ncolb

Input: $ncolb$, the number of columns of B .
 When **ncolb** = 0 then an immediate return is effected.
 Constraint: **ncolb** $\geq$ 0.

b[m][tdb]

Input: the leading m by $ncolb$ part of the array **b** must contain the matrix to be transformed.
 Output: **b** is overwritten by the transformed matrix.

tdb

Input: the second dimension of the array **b** as declared in the function from which nag_real_apply_q is called.
 Constraint: **tdb** $\geq$ **ncolb**.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_BAD_PARAM

On entry, parameter **trans** had an illegal value.
 On entry, parameter **wheret** had an illegal value.

NE_2_INT_ARG_LT

On entry, **m** = $\langle value \rangle$ while **n** = $\langle value \rangle$. These parameters must satisfy **m** $\geq$ **n**.
 On entry, **tda** = $\langle value \rangle$ while **n** = $\langle value \rangle$. These parameters must satisfy **tda** $\geq$ **n**.
 On entry, **tdb** = $\langle value \rangle$ while **ncolb** = $\langle value \rangle$. These parameters must satisfy **tdb** $\geq$ **ncolb**.

NE_INT_ARG_LT

On entry, **n** must not be less than 0: **n** = $\langle value \rangle$.
 On entry, **ncolb** must not be less than 0: **ncolb** = $\langle value \rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

The approximate number of floating-point operations is given by $2n(2m - n)ncolb$.

6.1. Accuracy

Letting C denote the computed matrix $Q^T B$, C satisfies the relation

$$QC = B + E$$

where $\|E\| \leq c\epsilon\|B\|$, ϵ is the **machine precision**, c is a modest function of m and $\|\cdot\|$ denotes the spectral (two) norm. An equivalent result holds for the computed matrix QB . See also Section 6.1 of nag_real_qr (f01qcc).

6.2. References

Golub G H and Van Loan C F (1989) *Matrix Computations* (2nd Edn) Johns Hopkins University Press, Baltimore.
Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Clarendon Press, Oxford.

7. See Also

nag_real_qr (f01qcc)

8. Example

To obtain the matrix $Q^T B$ for the matrix B given by

$$B = \begin{pmatrix} 1.10 & 0.00 \\ 0.90 & 0.00 \\ 0.60 & 1.32 \\ 0.00 & 1.10 \\ -0.80 & -0.26 \end{pmatrix}$$

following the QR factorization of the 5 by 3 matrix A given by

$$A = \begin{pmatrix} 2.0 & 2.5 & 2.5 \\ 2.0 & 2.5 & 2.5 \\ 1.6 & -0.4 & 2.8 \\ 2.0 & -0.5 & 0.5 \\ 1.2 & -0.3 & -2.9 \end{pmatrix}.$$

8.1. Program Text

```
/* nag_real_apply_q(f01qdc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf01.h>

#define MMAX 20
#define NMAX 10
#define NCBMAX 5

main()
{
    Integer tda = NMAX;
    Integer tdb = NCBMAX;

    double zeta[NMAX], a[MMAX][NMAX], b[MMAX][NCBMAX];
    Integer i, j, m, n, ncolb;

    Vprintf("f01qdc Example Program Results\n");
    Vscanf("%*[^\\n]"); /* skip headings in data file */
}
```

```

Vscanf(" %*[\n]");

Vscanf("%ld%ld", &m, &n);
if (m > MMAX || n > NMAX)
{
    Vprintf("m or n is out of range.\n");
    Vprintf("m = %2ld, n = %2ld\n", m, n);
}
else
{
    Vscanf(" %*[\n]");
    for (i = 0; i < m; ++i)
        for (j = 0; j < n; ++j)
            Vscanf("%lf", &a[i][j]);
    Vscanf(" %*[\n]");
    Vscanf("%ld", &ncolb);
    if (ncolb > NCBMAX)
    {
        Vprintf("ncolb is out of range.\n");
        Vprintf("ncolb = %2ld\n", ncolb);
    }
    else
    {
        Vscanf(" %*[\n]");
        for (i = 0; i < m; ++i)
            for (j = 0; j < ncolb; ++j)
                Vscanf("%lf", &b[i][j]);

        /* Find the QR factorization of A */
        f01qcc(m, n, (double *)a, tda, zeta, NAGERR_DEFAULT);

        /* Form Q'*B */
        f01qdc(Transpose, Nag_ElementsSeparate, m, n, (double *)a, tda, zeta,
              ncolb, (double *)b, tdb, NAGERR_DEFAULT);

        Vprintf("Matrix Q'*B\n");
        for (i = 0; i < m; ++i)
        {
            for (j = 0; j < ncolb; ++j)
                Vprintf(" %8.4f", b[i][j]);
            Vprintf("\n");
        }
    }
}
exit(EXIT_SUCCESS);
}

```

8.2. Program Data

f01qdc Example Program Data

Values of m and n.

5 3

Matrix A

2.0	2.5	2.5
2.0	2.5	2.5
1.6	-0.4	2.8
2.0	-0.5	0.5
1.2	-0.3	-2.9

Value of ncolb

2

Matrix B

1.1	0.0
0.9	0.0
0.6	1.32
0.0	1.1
-0.8	-0.26

8.3. Program Results

```
f01qdc Example Program Results
Matrix  Q'*B
-1.0000  -1.0000
-1.0000   1.0000
-1.0000  -1.0000
-0.1000   0.1000
-0.1000  -0.1000
```
